Journal of Advanced Mechanical Design, Systems, and Manufacturing

Vol. 1, No. 2, 2007

Optimization of Operation Sequence in CNC Machine Tools Using Genetic Algorithm*

Jaber ABU QUDEIRI**, HidehikoYAMAMOTO** and Rizauddin RAMLI**

**Intelligent Manufacturing Systems Laboratory, Gifu University

1-1 Yanagido, Gifu Shi, 501-1193, Japan

k3812203@guedu.cc.gifu-u.ac.jp

Abstract

The productivity of machine tools is significantly improved by using microcomputer based CAD/CAM systems for NC program generation. Currently, many commercial CAD/CAM packages that provide automatic NC programming have been developed and applied to various cutting processes. Many cutting processes machined by CNC machine tools. In this paper, we attempt to find an efficient solution approach to determine the best sequence of operations for a set of operations that located in asymmetrical locations and different levels. In order to find the best sequence of operations that achieves the shortest cutting tool travel path (CTTP), genetic algorithm is introduced. After the sequence is optimized, the G-codes that use to code for the travel time is created. CTTP can be formulated as a special case of the traveling salesman problem (TSP). The incorporation of genetic algorithm and TSP can be included in the commercial CAD/CAM packages to optimize the CTTP during automatic generation of NC programs.

Key words: NC Program, Tool Path, Operations Sequence, Genetic Algorithm, TSP.

1. Introduction

The productivity of machine tools can be significantly improved by using microcomputer based CAD/CAM systems for NC program generation. Currently, a number of commercial CAD/CAM packages that provide automatic NC programming have been developed and applied to various cutting processes. To cut any process using CNC machine tools, a tool path for the cutting tool should be determined. The total production time to cut any part using CNC machine tools consists of travel time (the time to move the CNC machine spindle between operations), switch time (the time to change the cutting tool for next operation), and the cutting time (the time that the cutting tool moves with cutting speed in air or in material). A survey of the literature shows that much research has been done on minimizing the cutting time[(1), (2)]; however, there is a lack of literature that studies the travel time between the operations. In order to minimize the travel time, the cutting tool travel path (CTTP) between operations should be minimized. CTTP can be classified into two types, one is continuous travel path. In this type the start point and the end point of each operation are the same[(3)], and it mainly appears in hole-cutting operations such as drilling, reaming, and tapping. The other is the discontinuous travel path, where the start and end points of the operation are different. This type includes most other operations such as grooving, pocketing, etc. Kolahan et al.[(4)] used a tabu-search approach to minimize the total processing cost for hole-making operations. They considered tool travel time, tool switching time and the cutting time and used the tabu-search algorithm to find the solution. They studied the holes that have the same surfaces level, which means the cutting tool, travels in *xy* plane to move from one hole to another. We have previously proposed optimization

*Received 4 Dec., 2006 (No. 06-0241)
[DOI: 10.1299/jamdsm.1.272]

method for hole-cutting operations sequence in CNC machine tools using genetic algorithm (GA)[3]. In this previous study, the operations contained a similar holes located in different levels. In contrast, our study solves CTTP for a different operation processes that can be done by using single cutting tool.

CTTP can be formulated as a special case of the traveling salesman problem (TSP) but is more complicated since the operations start and end points are different also are located in different planes and the cutting tool uses a save path to travel between such points. In this paper, we attempt to find an efficient solution approach to determine the best sequence of operations for the different process that achieves the shortest CTTP. In order to do this, GA is used to find the shortest CTTP. The G-codes needed for this operation are then created. Before going through the optimization of CTTP and the G-code generation, the basic concepts of the NC program configuration are explained in the following section.

2. CNC Program Configuration

For any operation using CNC machine tools, NC program is needed. There are two NC program types. One type can be coded by a programmer which is suitable for simple cutting process. The other type is generated by one of the commercial CAD/CAM packages that provide automatic NC programming, and this type is suitable for complex shape that cannot be programmed easily. NC program is a series of coded instructions required to produce a part. It controls the movement of the machine tool and the on/off control of auxiliary functions such as spindle rotation and coolant. The coded instructions are composed of letters, numbers and symbols and are arranged in a format of functional blocks. Each block consists of one or more NC words. An NC word is a collection of characters that begins with an address followed by numbers. A character is one byte (8 bits) of information representing a number, letter, or symbol. The sequence and arrangement of NC words in part programs must follow a certain pattern that is called the tape format or program. For any program, various functions and their related commanding methods are required. There are two main functions: (1) first function is a preparatory function which is designated in a program by the word address G, and (2) Miscellaneous functions which use the address letter M[5], An example of whole program can be shown in Fig. 1[6], the most important part of the program is a machining program parts which contain the G-codes needed to complete specific operation process.

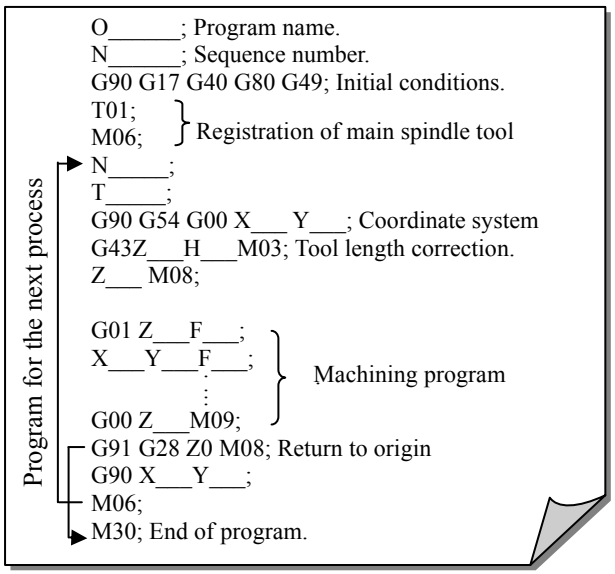

Fig.1 Example of NC program

3. Problem Description and Formulation

With the hand programing and the CAD/CAM packages that provide automatic NC programming, NC program does not use optimal techniques to find CTTP. The traditional processes to produce any products using CNC machines are: draw a product → find the tool path → create the NC program. Using this pattern the shortest CTTP couldn't be considered. It may take long machining time to complete the operations. In order to overcome this kind of disadvantage, we introduce GA. GA can efficiently find the sequence of operations that achieves the shortest CTTP. The cutting tool should travel through the safe CTTP to avoid damaging the workpiece and the cutting tool. In order to achieve the safe CTTP, the cutting tool must pass through an imaginary point when it moves between two operations in different planes.

The CTTP can be formulated as a special case of the traveling salesman problem (TSP). Cities in our problem are represented by operations and the salesman is represented by cutting tool. Similar to the TSP, CTTP consists of a number of nodes, with the distances between them given. The goal is to find the minimum length tour that visits each node exactly once. There are three main differences between CTTP and the conventional TSP. The first difference is that the arrival and departure points are assumed to be the same point in conventional TSP where the two points are different in CTTP, as shown in Fig. 2. The second difference is that the salesman in the TSP is returning to the starting city, where the tool in CTTP completes its tour when it reaches the last operation. The third difference is that the travel between the nodes in CTTP has some constraints to avoid any touch between the cutting tool and the workpiece.

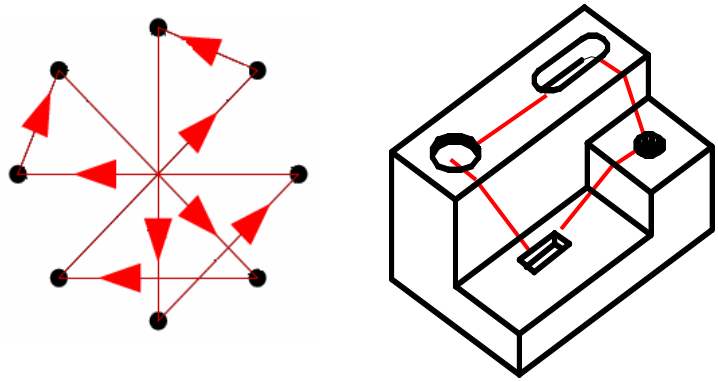

Fig. 2 Conventional TSP path and CTTP

4. Review of the TSP and its Variants

4.1 Traveling Salesman Problem (TSP)

TSP is one of the most widely studied combinatorial optimization problems[7), 8), 9)]. The most common practical interpretation of the TSP is that of a salesman seeking the shortest tour through N cities, visiting each city exactly once and returning back to the starting city. The problem statement is deceptively simple, but it still remains as one of the most challenging problems in operational research. To solve such problems of any size is by enumerating each possible tour and searching for the tour with the shortest path. Each possible tour is a permutation of *1, 2, 3, . . ., N*. where N is the number of cities, so the number of tours is $n!$. When n is large, it becomes impossible to find the path of every tour in polynomial time.

The TSP can be described mathematically as follows. Consider a complete digraph $G = (V,P)$ where V is the vertex set that represents the cities, $V = \{1,...,N\}$, A is the arc set, $A = \{(i, j), i, j \in V\}$ and D_{ij} is the distance between cities i and j. The TSP then can be solved by minimizing the total trip given by Eq. (1) below.

$$\text{Minimize} \quad \sum_{(i,j)\in A} D_{ij}\lambda_{ij} \quad \text{subjected to:} \tag{1}$$

$$\lambda_{ij} \in \{0,1\}$$

$$\sum_{i\in V} \lambda_{ij} = 1 \qquad j \in V \tag{2-A}$$

$$\sum_{j\in V} \lambda_{ij} = 1 \qquad i \in V \tag{2-B}$$

Constraints 2-A and 2-B ensure that each city is visited exactly once. The TSP is an NP-complete problem [10], heuristic algorithms that have relatively short running times and often give solutions that differ only slightly from the optimal solution. Several methods, usually based on search heuristics, have been applied for the traveling salesman problem, such as Greedy Algorithms[11], The Nearest Neighbor algorithm[12], A minimum spanning tree[11), (12)], simulated annealing[13], tabu search[14], neural networks[15]. One of the methods for solving the TSP is genetic algorithms (GA)[16]. GA is global optimization techniques using various optimization problems, such as the TSP[17].

4.2 Distances Between CTTP Nodes

To find the CTTP, the importance is how to find the distance between two pair of operations in the CTTP. Calculating the distance between two nodes depends on the plane where the two nodes belong to. Also depend on the start and end points of the operation. The distance D between a pair of operations through the CTTP can be calculated as follows:

CASE 1: If the two nodes belong to the same plane,

Then D can be calculated using Eq. (3).

$$D = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{3}$$

CASE 2: If the two nodes belong to different planes, as shown in Fig. 3, Then D can be calculated as follows.

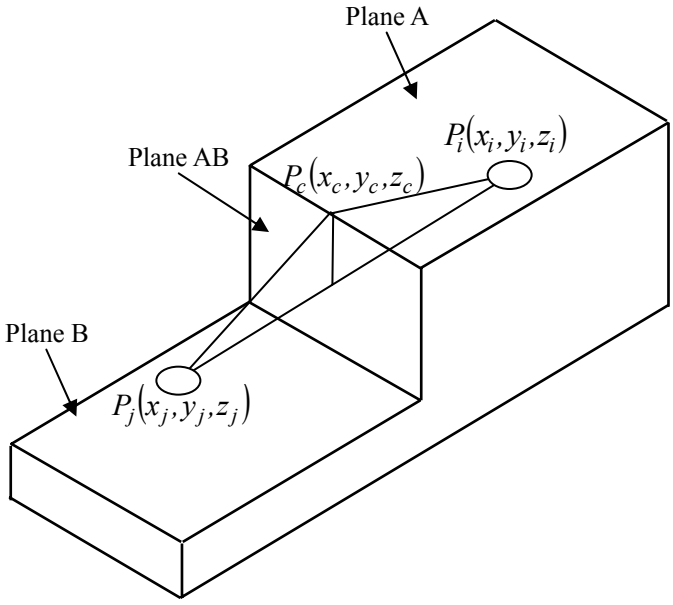

Fig. 3 Nodes in different planes

Refer to Fig. 3 and consider the points $P_i(x_i, y_i, z_i)$ and $P_j(x_j, y_j, z_j)$ which are located in plane A and plane B respectively. To move the tool path safely between the points P_i and P_j, the tool path should pass imaginary point (point P_c in Fig. 3). Thus, the distance D between points P_i and P_j can be calculated using the Eq. (4).

$$D = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2} + \sqrt{(x_j - x_c)^2 + (y_j - y_c)^2 + (z_j - z_c)^2} \quad (4)$$

The location of the imaginary point P_c can be determined using the following steps.

Step 1: Read the location of the work reference point.

Step 2: Read 3 points in plane *AB*, such as $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$ and $P_3(x_3, y_3, z_3)$

Step 3: Find the equation of plane *AB*.

The plane passing through three points P_1, P_2 and P_3 can be determined by carrying out the following rule.

A.Use points P_1, P_2 and P_3 to find vectors $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_1P_3}$ by using Eq.s (5) and (6) respectivly.

$$\overrightarrow{P_1P_2} = \langle x_2 - x_1, y_2 - y_1, z_2 - z_1 \rangle \quad (5)$$

$$\overrightarrow{P_1P_3} = \langle x_3 - x_1, y_3 - y_1, z_3 - z_1 \rangle \quad (6)$$

B. Find a normal vector $\vec{n}$ to the plane *AB*.

The normal vector to the plane is the cross product of vectors $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_1P_3}$. The normal vector $\vec{n}$ can be determined by using Eq. (7).

$$\vec{n} = \overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} = \begin{vmatrix} (y_2 - y_1) & (z_2 - z_1) \\ (y_3 - y_1) & (z_3 - z_1) \end{vmatrix} \vec{i} - \begin{vmatrix} (x_2 - x_1) & (z_2 - z_1) \\ (x_3 - x_1) & (z_3 - z_1) \end{vmatrix} \vec{j} + \begin{vmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_1) & (y_3 - y_1) \end{vmatrix} \vec{k} \quad (7)$$

where

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

C. Use P_1 and $\vec{n}$ to find the equation of the plane *AB* as follows.

$$A(x - x_1) + B(y - y_1) + C(z - z_1) = 0 \quad (8)$$

where

$A = (y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1)$,

$B = (x_2 - x_1)(z_3 - z_1) - (z_2 - z_1)(x_3 - x_1)$ and

$C = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$.

Step 4: Find the equation of line $\overline{P_iP_j}$ that passes through the points P_i and P_j. The line passing through the two points P_i and P_j can be determined by carrying out the following rule.

A. Find a vector parallel to the line by using the vector between the two points as given in Eq. (9).

$$\overrightarrow{P_iP_j} = (x_j - x_i)\vec{i} + (y_j - y_i)\vec{j} + (z_j - z_i)\vec{k} \quad (9)$$

B. Use the components of $\overrightarrow{P_iP_j}$ and P_i to express the parametric equation of the line by using Eq. (10).

$$\begin{aligned} x &= x_i + (x_j - x_i)t; \\ y &= y_i + (y_j - y_i)t; \\ z &= z_i + (z_j - z_i)t \end{aligned} \quad (10)$$

C. Solve t in each of *x*, *y* and *z* in Eq. (10) to find the symmetric as given in Eq. (11).

$$\frac{x - x_i}{xj - x_i} = \frac{y - y_i}{y_j - y_i} = \frac{z - z_i}{z_j - z_i} \quad (11)$$

Step 5: Find the intersection point $P_{int}(x_{int}, y_{int}, z_{int})$ between the plane *AB* and the line $\overline{P_iP_j}$ by carrying out the following rule.

A. Substitute the intersection point $P_{int}(x_{int}, y_{int}, z_{int})$ into the equation of the plane *AB* [Eq. (8)] as given in Eq. (12).

$$A(x_{int} - x_1) + B(y_{int} - y_1) + C(z_{int} - z_1) = 0 \quad (12)$$

B. Substitute the intersection point $P_{int}(x_{int}, y_{int}, z_{int})$ into the equation of the line $\overline{P_i P_j}$ [Eq. (10)] as given in Eq. (13).

$$\begin{aligned} x_{int} &= x_i + (x_j - x_i)t_{int}; \\ y_{int} &= y_i + (y_j - y_i)t_{int}; \\ z_{int} &= z_i + (z_j - z_i)t_{int} \end{aligned} \quad (13)$$

C. Combine Eq.s (12) and (13) to find t_{int}.

D. Substitute t_{int} into Eq. (13) to find x_{int}, y_{int} and z_{int}.

Step 6: Find the x, y and z coordinates of point P_c as follows.

$$\begin{aligned} x_c &= x_{int}, \\ y_c &= y_{int} \quad \text{and} \\ z_c &= z_i \end{aligned} \quad (14)$$

5 Genetic Algorithm

GA has been successfully applied to optimization problems in diverse fields. GA differ from other search techniques which depend on natural genetic evaluation process, start with an initial set of solutions selected randomly called *population* and follow the biological evolution process to improve upon them. The solution set in the population, called as *chromosome* or *individual,* represents a solution to the optimization problem. Each individual contains a number of *genes.* The individuals in the initial population are *evaluated* to measure its *fitness.* Based on the individuals' fitness, some individuals are selected from the current population as *parents.* GA operations like crossover and mutation are then applied to the parents to generate new individuals, called *offspring's*. These offspring's are then evaluated. A new population can be generated by selecting some of the parents and offspring's. The iteration, called a *generation* will continue until the fitness reaches its maximum value. The best overall solution becomes the candidate solution to the problem. Similar cutting operations (holes) are considered in the GA operations. The operations for our GA are described in §§ 5.1 ~ 5.4.

5.1 Encoding

The CTTP, same as the TSP, is a sequential problem so we need to make some changes to the traditional genetic algorithm to solve this type of problem. To encode CTTP we use a path representation where the hole centers are listed in the order in which they are visited.

For example, assuming there are 4 holes A, B, C, D, if a cutting tool starts from the center of hole B, through hole centers C, D, A, the individual will be B C D A. For N holes, we initialize the population by randomly placing 1 to *N* into *N* length individual and guaranteeing that each hole center appears exactly once. Then the individual for *N* holes can be coded as shown in Eq. (15).

$$\begin{aligned} individual &= [G_1\ G_2 \cdots\ G_i \cdots G_N], \\ G_i \cap [G_1\ G_2 \cdots\ G_i \cdots G_N] &= G_i\ \forall i = 1,2,\cdots N \end{aligned} \quad (15)$$

5.2 Crossover

The traditional crossover operation is not suitable for this type of representation because each gene should appear exactly once in each individual. There are several crossover operators for TSPs. We use Greedy Crossover which was invented by Grefenstette [18]. Greedy crossover selects the first hole center of one parent, compares the hole centers leaving that hole in both parents, and chooses the closer one to extend the tour. If one hole has already appeared in the tour, we choose the other city. If both holes have already appeared, we randomly select a non-selected hole. For example, consider the two parents as shown in Fig. 4. If the first individual is chosen as the template, child 1 can be generated using the following steps:

Step 1: Select hole 5 (the first hole in the iduvidual) as the first city of child 1.

Step 2: Find the edges after gene 5 in both parents: (5, 2) and (5, 6) and compare the distance of

these two edges. The hole located in the shorter distance will be chosen as the second gene, assuming the distance between hole 5 and hole 2 is shorter, we select hole 2 as the next gene of child 1.

Step 3: Find the edges after hole 2: (2, 4) and (2, 3). If the distance between hole 2 and hole 3 is shorter, we select hole 3 as the next gene.

Step 4: Find the edges after hole 3: (3, 1) and (3, 4). If the distance between hole 3 and hole 1 is shorter, we select hole 1 as the next gene.

Step 5: Find the edges after city 1: (1, 6) and (1, 2), since hole 2 appears in child 2, we select hole 6 as the next gene.

Step 6: Find the edges after city 6: (6, 5) and (6, 1), but hole 5 and hole 1 both appear in child 1. We select a non-selected hole, which is hole 4, and thus produce a legal child. We can use the same procedure to generate child 2 as shown in Fig. 4. After crossover, both offspring encode legal tours.

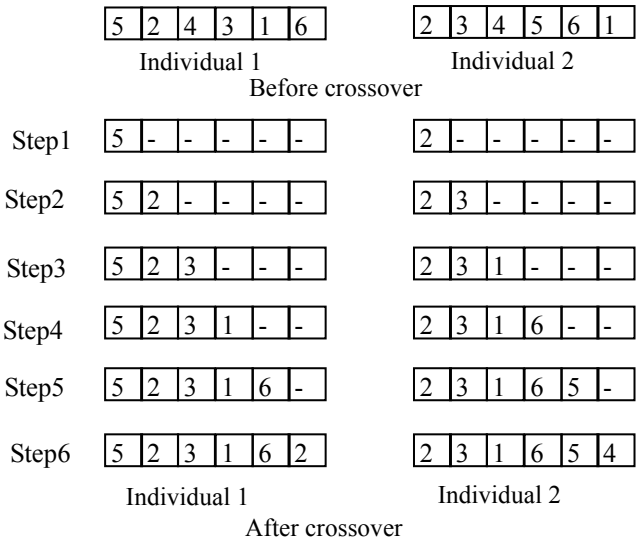

Fig. 4 Greedy crossover procedure

5.3 Mutation

For the same reason that we do not use the traditional crossover operator, we can not use the traditional mutation operator. Instead of using the traditional mutation operator, we randomly select two genes in one individual and *swap* their values. Thus, we still have legal tours after swap mutation as shown in Fig. 5.

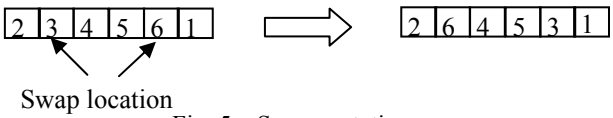

Fig. 5 Swap mutation

5.4 Fitness Evaluations

Following the TSP procedure the evaluation function for the N holes is the sum of distances between every pair of hole centers in the tour. The distance between each pair of the tour can be calculated as described in § 4.2.

6. Generation the G-Code for the CTTP

The G-Code to operate a set of holes located in different levels can be coded using the following algorithm.

Step 1: Read the coordinate of each node of the optimized CTTP.

Step 2: Code the traverse motion command **G00** then the *X, Y* and *Z* coordinates of the first hole in the CTTP.
Step 3: Read the first hole cutting cycle.
Step 4: Check the current and next node planes.
Step 5: If the current and next node related to the same plane, **code** the traverse motion command **G00** then the *X, Y* and *Z* coordinates of the next hole in the CTTP. Otherwise (If the current and next node related to the different planes), find the imaginary point coordinates.
Step 6: Code **G00** then the *X, Y* and *Z* coordinates of the imaginary point, then, the *X, Y* and *Z* coordinates of the next hole in the CTTP.
Step7: Repeat Steps 4-6 until the cutting tool reaches the last hole in the CTTP.

7. Algorithms Performance

The improvement of the solution was measured with respect to an initial random solution. For 84 population size problem, the algorithm resulting in 240% improved over an initial random solution. Fig. 6 shows the percentage improvements for problem with population size achieved by the algorithm. The improvement in solution is increased by increasing the number of holes.

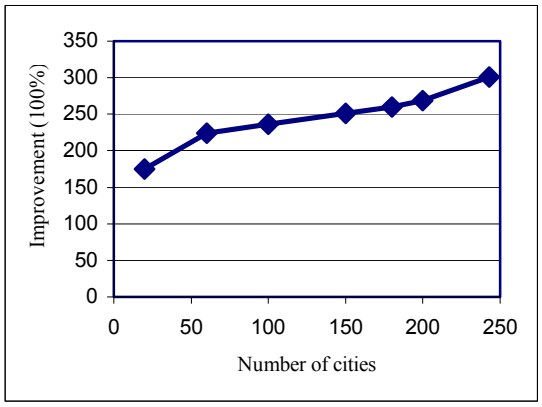

Fig. 6 CTTP improvements

8. Application Examples

8.1 Example 1

The TSP and GA have been incorporated to find the shortest CTTP to operate the holes in Fig. 7.

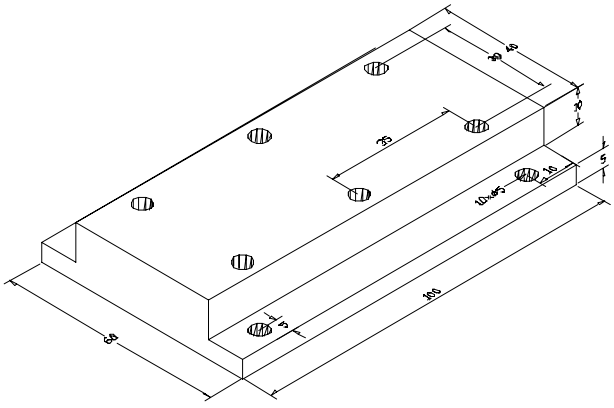

Fig. 7 Application example 1

After a number of generations the shortest CTTP for Fig. 7 example is as shown in Fig. 8. The shortest tool path is 255.2432 mm, where the conventional method that adopted plane by plane cutting needs 370 mm.

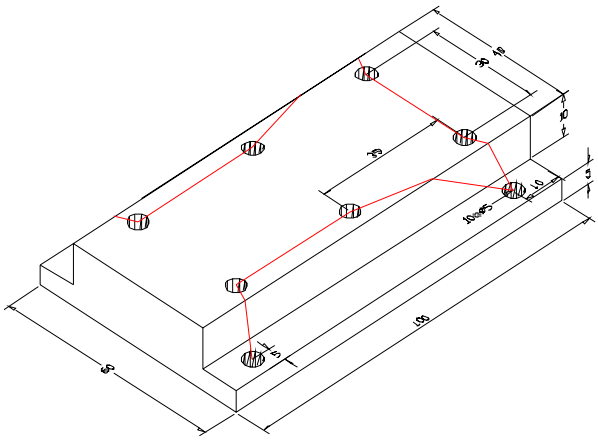

Fig. 8 Shortest CTTP of application example 1

8.2 Example 2

The TSP and GA have been incorporated to find the shortest CTTP to operate the holes in Fig. 9. Figure 9 shows 243 holes located in 3 planes. 33 holes, 111 holes and 99 holes located in planes A, B and C respectively, this application is not practical one, it is only to show the effective of this technique for CTTP optimization.

All dimensions in mm

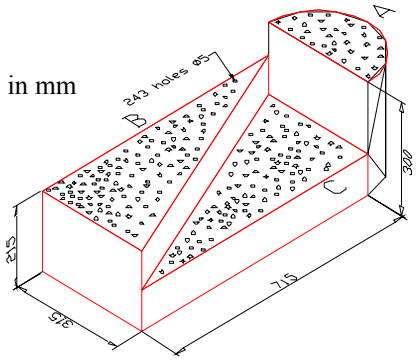

Fig. 9 Application example 2

First, we describe the shortest CTTP. As a result of GA, the operation sequence achieves the shortest CTTP is found. Figure 10 shows the CTTP for the final individual.

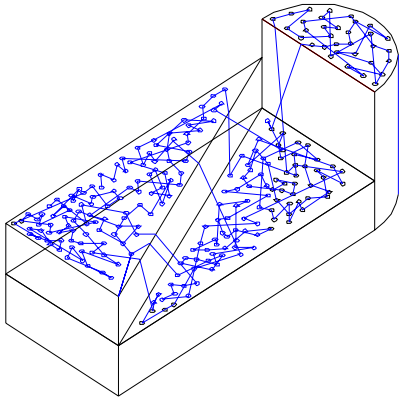

Fig. 10 Shortest CTTP for 243 holes example 2

Figure 11 shows the best fitness curve with the generation. In this curve, the fitness decreases with the generations until it reaches its minimum value. The fitness is improved approximately 300% from the initial generation to the final generation. That means the incorporation of GA and TSP is useful for the CTTP optimization.

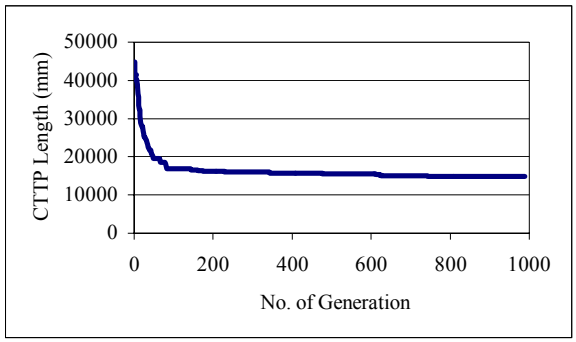

Fig. 11 Fitness curve

Second, we described the G-code generation for the optimized CTTP. Figure 12 shows some commands of the G-code.

```
:
:
G00 X 212.8612; Y-55.57369;   Z-84.0979;
First hole cutting cycle;
G00 X 237.5980;   Y-26.9644;   Z-84.0979;
Second hole cutting cycle;
G00 X245.4096;   Y -22.6297;   Z-84.0979;
:
G00 X 99.1587;   Y-57.3074;   Z-84.0979;
Last hole cutting cycle;
:
:
```

Fig. 12. Some commands of G-Code

9. Conclusions

This paper finds the efficient sequence of operations located in asymmetrical locations and different levels that achieves the shortest CTTP. The cutting tool travel path was formulated as a special case of the TSP with some constraints. The main constraint is forcing the cutting tool to visit an imaginary point to avoid any touch with the workpiece. The incorporation of GA and TSP introduced in this paper is used to find the sequence of operation. This incorporation of GA and TSP is not restricted to machining but can be applied to any similar problem such as spot welding sequence. The percentage improvements from the initial to the final generation depend on the problem size. As the number of holes increases, the improvement is increased. The incorporation of GA and TSP can be included in the commercial CAD/CAM packages to optimize the CTTP during automatic generation of CNC machine programs

References

(1) Castelino, K., D'Souza, R. and Wright, P. K., "Toolpath optimization for minimizing airtime during machining", *Journal of Manufacturing Systems*, Vol. 22, No. 3, pp173-180, 2003.
(2) D'Souza, R., Wright, P. K. and Sequin, C.H., "Automated microplanning for 2.5 D pocket machining", *Journal of Manufacturing Systems*, Vol. 20, No. 4, pp288-296, 2001.
(3) Jaber E. Abu Qudeiri, Al-Momani Raid, Mohamed Anouar Jamali and HidehikoYamamoto, Optimization Hole-Cutting Operations Sequence in CNC Machine Tools Using GA,

International Conference on Service systems and Service Management (ICSSM'06),CD, pp.501 – 506, Troyes – France, 2006.

(4) Kolahan, F., and Liang, M. "Optimization of hole-making operations: a tabu-search approach", *International Journal of Machine Tools & Manufacture,* Vol. 40, No. 12, pp1735–1753, 2000.

(5) Lin, S.C. and Shiue, F.C., "Mastercam Version 7", *Scholar's international publishing corp.*, Ann Arbor, Michigan, 1998.

(6) Miyamoto, K., Takada, Y., Shimamora Y., "Machining Centers: Numerically controlled series machine tool", *published by Overseas Vocational Training Association,* 1994.

(7) Garfinkel, R. and Gilbert, K., "Minimizing wallpaper waste, Part 1: A class of traveling salesman problems", *Operational Research*, Vol. 25, No. 5, pp741-751, 1977.

(8) Langevin A., Soumis, F. and Desrosiers, J., "Classification of traveling salesman problem formulations", *Operations Research Letters,* Vol. 9, pp127-132, 1990.

(9) Laporte, G., "The Traveling salesman problem: An overview of exact and approximate algorithms", *European Journal of Operations Research*, Vol. 59, No. 2, pp231-247. 1992.

(10) Garey, M.R., and Johnson, D.S., "Computers and Intractability: A guide to the theory of NP-completeness", *W.H. Freeman & Co Ltd*, New York, 1979.

(11) Lawler, E.L., Lenstra, J.K., Rinnooy, A.H.G. and Shmoys., D.B., "The traveling salesman problem", *John Wiley and Sons*, New York, 1985.

(12) Reinelt, G., "The Traveling salesman: Computational solutions for TSP applications", *Springer Verlag*, Berlin, 1994.

(13) Laarhoven van, P. J. and Aarts, E.H., "Simulated annealing: Theory and applications", D. Reidel, Dordrecht, The Nethelands, 1987.

(14) Fiechter, C. N., "A parallel tabu search algorithm for large traveling salesman problems", *Discrete Applied Mathematics*, Vol. 51, No. 3, pp243-267, 1994.

(15) Aarts, E.H.L. and Stehouwer, H.P., "Neural networks and the travelling salesman problem", Proc. Intl. Conf. on Artificial Neural Networks, p950-955, *Springer-Verlag*, London, 1993.

(16) Merz, P., Freisleben, B., "Genetic Local Search for the TSP: New Results", *proceeding of IEEE International Conference on Evolutionary Computation (ICEC '97)*, University Place Hotel Indianapolis, 1997.

(17) Martin, O., Otto, S.W., and Felten, E.W. "Large-Step markov chains for the traveling salesman problem", *Complex Systems*, Vol.5, pp299-326, 1991.

(18) Grefenstette, B J., Gopal, R., Rosmaita, R. and Gucht, D., "Genetic algorithms for the traveling salesman problem", *In Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Eribaum Associates, NJ, 1985.